



(21)申請案號：108101178

(22)申請日：中華民國 108 (2019) 年 01 月 11 日

(51)Int. Cl. : **G06F9/38 (2006.01)**

(30)優先權：2018/03/30 中國大陸 201810278922.7

(71)申請人：香港商阿里巴巴集團服務有限公司(香港地區) ALIBABA GROUP SERVICES
LIMITED (HK)
香港

(72)發明人：曹玉斌(CN)

(74)代理人：林志剛

(56)參考文獻：

TW I561980B

TW 201714085A

CN 106874168A

CN 107145429A

US 8850394B2

審查人員：蘇齊賢

申請專利範圍項數：10 項 圖式數：3 共 32 頁

(54)名稱

函式選取方法和伺服器

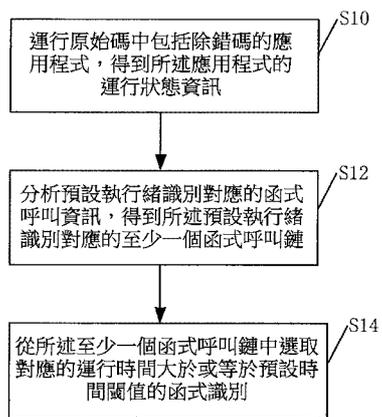
(57)摘要

本說明書實施例提供一種函式選取方法和伺服器。所述方法包括：運行原始碼中包括除錯碼的應用程式，得到所述應用程式的運行狀態資訊；所述運行狀態資訊包括執行緒識別對應的函式呼叫資訊；所述函式呼叫資訊包括函式識別以及函式識別對應的時間戳；分析預設執行緒識別對應的函式呼叫資訊，得到所述預設執行緒識別對應的至少一個函式呼叫鏈；每個所述函式呼叫鏈包括至少一個函式識別；從所述至少一個函式呼叫鏈中選取對應的運行時間大於或等於預設時間閾值的函式識別；函式識別對應的運行時間基於該函式識別對應的時間戳得到。

指定代表圖：

符號簡單說明：

無



【圖 2】

【發明說明書】

【中文發明名稱】

函式選取方法和伺服器

【技術領域】

本說明書實施例是關於電腦之技術領域，特別是關於一種函式選取方法和伺服器。

【先前技術】

在應用程式的生命週期中，當一個應用程式自身發生碼升級、版本迭代或者漏洞修復等變化時，或者，與該應用程式相關聯的其它應用程式發生碼升級、版本迭代或者漏洞修復等變化時，通常需要定位出該應用程式中的耗時函式，以便對該應用程式的回應時間進行優化。所述耗時函式可以為運行時間較長的函式。所述回應時間可以包括啟動時間、執行某一功能的時間等。

在相關技術中，可以獲取應用程式的追蹤檔案（`trace` 檔案），並可以採用人工的方式來分析所述追蹤檔案以從中識別出耗時函式。例如開發人員可以借助 `TraceView`（一種測試應用程式性能的工具）來分析所述追蹤檔案以從中識別出耗時函式。但是，採用人工的方式來識別耗時函式，效率和準確性較低，難以滿足實際的需求。

【發明內容】

本說明書實施例的目的是提供一種函式選取方法和伺服器，以提高耗時函式識別的效率和準確性。

為實現上述目的，本說明書實施例提供一種函式選取方法，包括：運行原始碼中包括除錯碼的應用程式，得到所述應用程式的運行狀態資訊；所述運行狀態資訊包括執行緒識別對應的函式呼叫資訊；所述函式呼叫資訊包括函式識別以及函式識別對應的時間戳；分析預設執行緒識別對應的函式呼叫資訊，得到所述預設執行緒識別對應的至少一個函式呼叫鏈；每個所述函式呼叫鏈包括至少一個函式識別；從所述至少一個函式呼叫鏈中選取對應的運行時間大於或等於預設時間閾值的函式識別；函式識別對應的運行時間基於該函式識別對應的時間戳得到。

為實現上述目的，本說明書實施例提供一種伺服器，包括：運行單元，用於運行原始碼中包括除錯碼的應用程式，得到所述應用程式的運行狀態資訊；所述運行狀態資訊包括執行緒識別對應的函式呼叫資訊；所述函式呼叫資訊包括函式識別以及函式識別對應的時間戳；分析單元，用於分析預設執行緒識別對應的函式呼叫資訊，得到所述預設執行緒識別對應的至少一個函式呼叫鏈；每個所述函式呼叫鏈包括至少一個函式識別；選取單元，用於從所述至少一個函式呼叫鏈中選取對應的運行時間大於或等於預設時間閾值的函式識別；函式識別對應的運行時間基於該函式識別對應的時間戳得到。

為實現上述目的，本說明書實施例提供一種伺服器，

包括：記憶體，用於儲存電腦指令；處理器，用於執行所述電腦指令實現以下步驟：運行原始碼中包括除錯碼的應用程式，得到所述應用程式的運行狀態資訊；所述運行狀態資訊包括執行緒識別對應的函式呼叫資訊；所述函式呼叫資訊包括函式識別以及函式識別對應的時間戳；分析預設執行緒識別對應的函式呼叫資訊，得到所述預設執行緒識別對應的至少一個函式呼叫鏈；每個所述函式呼叫鏈包括至少一個函式識別；從所述至少一個函式呼叫鏈中選取對應的運行時間大於或等於預設時間閾值的函式識別；函式識別對應的運行時間基於該函式識別對應的時間戳得到。

由以上本說明書實施例提供的技術方案可見，本說明書實施例中，持續整合伺服器可以運行原始碼中包括除錯碼的應用程式，得到所述應用程式的運行狀態資訊；可以分析預設執行緒識別對應的函式呼叫資訊，得到所述預設執行緒識別對應的至少一個函式呼叫鏈；可以從所述至少一個函式呼叫鏈中選取對應的運行時間大於或等於預設時間閾值的函式識別。本說明書實施例中，所述持續整合伺服器可以分析應用程式中預設執行緒識別對應的函式呼叫資訊，得到至少一個函式呼叫鏈；可以基於預設時間閾值從所述至少一個函式呼叫鏈中過濾出耗時函式。這樣，所述持續整合伺服器可以提高耗時函式識別的效率和準確性。

【圖式簡單說明】

為了更清楚地說明本說明書實施例或現有技術中的技術方案，下面將對實施例或現有技術描述中所需要使用的附圖作簡單地介紹，顯而易見地，下面描述中的附圖僅僅是本說明書中記載的一些實施例，對於本領域普通技術人員來講，在不付出創造性勞動性的前提下，還可以根據這些附圖獲得其他的附圖。

圖1為本說明書實施例一種函式選取系統的示意圖；

圖2為本說明書實施例一種函式選取方法的流程圖；

圖3為本說明書實施例一種伺服器的功能結構圖；以及

圖4為本說明書實施例一種伺服器的功能結構圖。

【實施方式】

下面將結合本說明書實施例中的附圖，對本說明書實施例中的技術方案進行清楚、完整地描述，顯然，所描述的實施例僅僅是本說明書一部分實施例，而不是全部的實施例。基於本說明書中的實施例，本領域普通技術人員在沒有作出創造性勞動前提下所獲得的所有其他實施例，都應當屬於本說明書保護的範圍。

請參閱圖1。本說明書實施例提供一種函式選取系統。所述函式選取系統可以包括版本伺服器、持續整合伺服器（Continuous Integration Server）和至少一個終端設備。

在本實施例中，所述終端設備可以為開發人員所使用的終端設備，例如可以為個人電腦（PC機）、或伺服器。開發人員可以使用所述終端設備來編輯應用程式的原始碼。所述版本伺服器可以用於儲存並管理應用程式各個版本的原始碼。例如，所述版本伺服器可以運行有SVN（Subversion）、CVS（Concurrent Version System）等應用程式版本控制系統；可以基於所述應用程式版本控制系統來管理應用程式各個版本的原始碼。所述持續整合伺服器可以用於對應用程式的原始碼進行編譯和測試。

在本實施例中，所述終端設備可以向所述版本伺服器上傳應用程式的原始碼。所述版本伺服器可以接收並儲存應用程式的原始碼。所述持續整合伺服器可以從所述版本伺服器中獲取應用程式的原始碼；可以編譯所述應用程式的原始碼，得到所述應用程式；可以對所述應用程式的性能進行測試，得到測試結果；可以向所述終端設備發送所述測試結果。所述終端設備可以接收所述測試結果。這樣，透過所述測試結果，開發人員可以從所述應用程式的原始碼中獲取影回應用程式性能的函式。這裡所述性能例如可以為啟動時間、或耗電量等；所述函式可以理解為能夠實現一定功能的電腦程式指令集合。

請參閱圖2。本說明書實施例提供一種函式選取方法。所述函式選取方法以持續整合伺服器為執行主體，可以包括以下步驟。

步驟S10：運行原始碼中包括除錯碼的應用程式，得

到所述應用程式的運行狀態資訊。

在本實施例中，所述應用程式可以為用於完成一項或多項工作的電腦程式。所述應用程式具體可以為任意類型的應用程式，例如可以為支付類型的應用程式、視頻播放類型的應用程式、或圖像處理類型的應用程式等。

在本實施例中，在應用程式的開發階段，開發人員可以在所述應用程式的原始碼中插入除錯碼，以便在運行所述應用程式時能夠得到所述應用程式的運行狀態資訊。在後續過程中透過分析所述運行狀態資訊，便可以發現所述應用程式原始碼中的缺陷。所述除錯碼可以包括除錯函式。所述除錯函式可以用於產生運行狀態資訊。所述除錯函式例如可以包括 `Debug.startMethodTracing`、和 `Debug.stopMethodTracing` 等。當然，所述除錯碼還可以包括其它的功能函式，例如檢測函式。所述檢測函式可以用於在檢測到滿足特定條件時觸發所述除錯函式，例如在檢測到根目錄存在某一檔案時觸發所述除錯函式。

在本實施例中，所述運行狀態資訊可以包括應用程式的至少一個執行緒的執行緒識別、以及執行緒識別對應的函式呼叫資訊等。所述執行緒識別可以用於識別執行緒，例如可以為執行緒的名稱、或編號等。所述函式呼叫資訊可以用於描述執行緒識別識別的執行緒所呼叫的函式。所述函式呼叫資訊具體可以包括至少一個函式識別、以及函式識別對應的時間戳等。所述函式識別可以用於識別函式，例如可以為函式的名稱、或編號等。所述時間戳可以

包括進入時間戳和退出時間戳。所述進入時間戳可以用於表示函式識別識別的函式的進入時刻；所述退出時間戳可以用於表示函式識別識別的函式的退出時刻。基於所述進入時間戳和所述退出時間戳能夠計算出函式識別識別的函式的運行時間。需要說明的是，有鑒於一個執行緒可以一次或多次呼叫一個函式，所述函式呼叫資訊中的函式識別可以對應一對或多對時間戳，每對時間戳可以包括一個進入時間戳和一個退出時間戳。

在本實施例的一個場景示例中，所述持續整合伺服器可以運行原始碼中包括除錯碼的應用程式，進而得到追蹤檔案（`trace`檔案）。所述追蹤檔案可以包括所述應用程式的運行狀態資訊。具體地，例如，所述追蹤檔案可以包括：

```
Trace (threadID action usecs class.method):
.....
12693  ent  16752  ....android.os.Handler.dispatchMessage;
.....
12622  ent  38450  .....android.os.MessageQueue.next;
.....
12693  xit  18976  ....android.os.Handler.dispatchMessage;
.....
12622  xit  49559  .....android.os.MessageQueue.next;
.....
```

上述追蹤檔案中；

`threadID`表示執行緒識別。例如，執行緒識別12622用於識別執行緒 `main`；執行緒識別12693用於識別執行緒

OnLineMonitor。

`class.method`表示函式識別。例如，`android.os.Handler.dispatchMessage`和`android.os.MessageQueue.next`分別為函式識別。

Action表示動作。例如，Action可以包括`ent`和`xit`。Ent表示進入函式；`xit`表示退出函式。

`usecs`表示時間戳。例如，16752為函式識別`android.os.Handler.dispatchMessage`對應的進入時間戳；18976為函式識別`android.os.Handler.dispatchMessage`對應的退出時間戳。

在本實施例的一個實施方式中，考慮到開發人員透過在應用程式原始碼的不同位置插入除錯碼，便可以得到所述應用程式在不同運行階段的運行狀態資訊。例如，開發人員透過在應用程式原始碼的啟動碼處插入除錯碼，便可以得到所述應用程式在啟動階段的運行狀態資訊。所述啟動碼可以為應用程式在啟動階段執行的碼。如此，所述運行狀態資訊可以包括所述應用程式在預設運行階段的運行狀態資訊。所述預設運行階段可以包括啟動階段。當然，所述預設運行階段還可以包括其它階段，例如執行某一功能的階段等。

在本實施例中，所述持續整合伺服器可以接收版本伺服器發來的應用程式的原始碼，所述應用程式的原始碼中可以包括除錯碼；可以基於所述應用程式的原始碼，產生所述應用程式；可以運行所述應用程式，得到所述應用程

式的運行狀態資訊。具體地，所述持續整合伺服器可以編譯所述應用程式的原始碼，得到所述應用程式。所述版本伺服器可以每間隔預設時間週期，向所述持續整合伺服器發送自身儲存的所述應用程式最新版本的原始碼。所述預設時間週期可以根據實際需要靈活設定，例如可以為10天、25天、或30天等。或者，有鑒於所述版本伺服器通常認為接收到的原始碼為所述應用程式最新版本的原始碼，所述版本伺服器可以在接到終端設備發來的應用程式的原始碼後，儲存並向所述持續整合伺服器發送所述應用程式的原始碼。

在本實施例的一個場景示例中，所述應用程式的原始碼中可以包括除錯碼。所述除錯碼可以包括檢測函式和除錯函式。所述檢測函式可以用於在檢測到根目錄存在某一檔案後觸發所述除錯函式，所述檔案例如可以為abc.txt。所述除錯函式可以包括 `Debug.startMethodTracing`、和 `Debug.stopMethodTracing`。具體地，所述除錯函式 `Debug.startMethodTracing` 可以位於所述應用程式原始碼中啟動碼的起始位置；所述除錯函式 `Debug.stopMethodTracing` 可以位於所述應用程式原始碼中啟動碼的結束位置。

在本場景示例中，開發人員在編輯完成應用程式的原始碼後，可以使用終端設備向版本伺服器發送所述應用程式的原始碼。所述版本伺服器可以接收並儲存所述應用程式的原始碼；可以向所述持續整合伺服器發送所述應用程

式的原始碼。所述持續整合伺服器可以接收所述應用程式的原始碼；可以編譯所述應用程式的原始碼，得到所述應用程式。所述持續整合伺服器可以在根目錄寫入預先約定的檔案（例如檔案 `abc.txt`）以便觸發所述應用程式原始碼中的檢測函式；可以運行所述應用程式，得到追蹤檔案。所述追蹤檔案可以包括所述應用程式在啟動階段的運行狀態資訊。

步驟 S12：分析預設執行緒識別對應的函式呼叫資訊，得到所述預設執行緒識別對應的至少一個函式呼叫鏈。

在本實施例中，所述預設執行緒識別的數量可以為一個或多個。所述預設執行緒識別識別的執行緒可以為所述應用程式的部分或全部執行緒。例如，所述預設執行緒識別識別的執行緒可以為所述應用程式的與啟動相關的執行緒。具體地，例如，所述預設執行緒識別可以包括執行緒識別 12622 和 12693，執行緒識別 12622 可以用於識別執行緒 `main`，執行緒識別 12693 可以用於識別執行緒 `OnLineMonitor`。所述函式呼叫鏈可以包括至少一個函式識別。每個所述函式呼叫鏈中的函式識別識別的函式間可以具有逐級呼叫關係。

在本實施例的一個場景示例中，某一函式呼叫鏈可以包括 16 個函式識別，具體包括：

```

com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run;
  java.lang.reflect.Method.invoke;
    android.app.ActivityThread.main;
      android.os.Looper.loop;
        android.os.Handler.dispatchMessage;
          android.os.Handler.handleCallback;
            com.alipay.mobile.nebulacore.bridge.H5BridgeImpl$2.run;
              com.alipay.mobile.nebulacore.bridge.H5BridgeImpl.access$100;
                com.alipay.mobile.nebulacore.bridge.H5BridgeImpl.executeWeb;
                  com.alibaba.fastjson.JSON.toJSONString;
                    com.alibaba.fastjson.serializer.JSONSerializer.write;
                      com.alibaba.fastjson.serializer.MapSerializer.write;
                        com.alibaba.fastjson.serializer.MapSerializer.write;
                          com.alibaba.fastjson.serializer.ListSerializer.write;
                            com.alibaba.fastjson.serializer.ListSerializer.write;
                              com.alibaba.fastjson.serializer.MapSerializer.write。

```

上述函式呼叫鏈中的函式識別識別的函式間可以具有逐級呼叫關係。具體地，函式識別

`com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run`識別的函式可以呼叫函式識別

`java.lang.reflect.Method.invoke`識別的函式；函式識別

`java.lang.reflect.Method.invoke`識別的函式可以呼叫函式

識別 `android.app.ActivityThread.main`識別的函式；依次類

推，函式識別 `com.alibaba.fastjson.serializer.ListSerializer.write`

識別的函式可以呼叫函式識別

`com.alibaba.fastjson.serializer.MapSerializer.write`識別的

函式。

在本實施例中，針對每個預設執行緒，所述持續整合伺服器可以獲取該預設執行緒在所述運行狀態資訊中對應的函式呼叫資訊；可以以獲取的函式呼叫資訊為目標函式呼叫資訊；可以分析所述目標函式呼叫資訊中各個函式識別識別的函式間的呼叫關係，得到該執行緒識別對應的至少一個函式呼叫鏈。

步驟 S14：從所述至少一個函式呼叫鏈中選取對應的運行時間大於或等於預設時間閾值的函式識別。

在本實施例中，函式識別對應的運行時間可以基於該函式識別對應的時間戳得到。具體地，如前所述，函式識別對應的時間戳可以包括進入時間戳和退出時間戳。如此，針對所述至少一個函式呼叫鏈中的每個函式識別，所述持續整合伺服器可以以該函式識別對應的退出時間戳表示的時刻為退出時刻；可以以該函式識別對應的進入時間戳表示的時刻為進入時刻；可以計算所述退出時刻和所述進入時刻之間的差值，作為該函式識別對應的運行時間。

在本實施例中，針對所述至少一個函式呼叫鏈中的每個函式呼叫鏈，所述持續整合伺服器可以查找該函式呼叫鏈中是否包括運行時間大於或等於所述預設時間閾值的函式識別；在該函式呼叫鏈中包括運行時間大於或等於預設時間閾值的函式識別時，可以從該函式呼叫鏈中選取對應的運行時間大於或等於所述預設時間閾值的函式識別。所述預設時間閾值可以根據實際需要靈活設定，例如可以為 100ms、150ms、或 180ms 等。

需要說明的是，有鑒於一個函式識別可以對應一對或多對時間戳，一個函式識別可以對應一個或多個運行時間。在函式識別對應的一個或多個運行時間中的任意一個運行時間大於或等於所述預設時間閾值時，所述持續整合伺服器便可以認為該函式識別對應的運行時間大於或等於所述預設時間閾值。

還需要說明的是，在一個函式的運行時間大於或等於所述預設時間閾值時，逐級呼叫直至該函式的其它各個函式的運行時間也均大於或等於所述預設時間閾值。如此，針對所述至少一個函式呼叫鏈中的每個函式呼叫鏈，所述持續整合伺服器從該函式呼叫鏈中選取的函式識別能夠形成該函式呼叫鏈的一個子函式呼叫鏈。

在本實施例的一個場景示例中，所述至少一個函式呼叫鏈可以包括函式呼叫鏈CA和CB。

在本場景示例中，函式呼叫鏈CA可以包括FA、FB、FC、FD、FE等5個函式識別。函式呼叫鏈CA中的函式識別識別的函式間可以具有逐級呼叫關係。具體地，函式識別FA識別的函式可以呼叫函式識別FB識別的函式；函式識別FB識別的函式可以呼叫函式識別FC識別的函式；依次類推，函式識別FD識別的函式可以呼叫函式識別FE識別的函式。在函式呼叫鏈CA中，函式識別FA、FB、FC對應的運行時間大於所述預設時間閾值。

在本場景示例中，函式呼叫鏈CB可以包括FC、FF、FG、FH、FI、FJ等6個函式識別。函式呼叫鏈CB中的函式

識別識別的函式間可以具有逐級呼叫關係。具體地，函式識別FC識別的函式可以呼叫函式識別FF識別的函式；函式識別FF識別的函式可以呼叫函式識別FG識別的函式；依次類推，函式識別FI識別的函式可以呼叫函式識別FJ識別的函式。在函式呼叫鏈CB中，函式識別FC、FF、FG、FH對應的運行時間大於所述預設時間閾值。

在本場景示例中，所述持續整合伺服器可以從函式呼叫鏈CA中選取函式識別FA、FB、FC。函式識別FA、FB、FC能夠形成函式呼叫鏈CA的一個子函式呼叫鏈。所述持續整合伺服器可以從函式呼叫鏈CB中選取函式識別FC、FF、FG、FH。函式識別FC、FF、FG、FH能夠形成函式呼叫鏈CB的一個子函式呼叫鏈。

在本實施例的一個實施方式中，所述持續整合伺服器還可以獲取選取的函式識別所對應的運行時間。

在本實施方式的一個場景示例中，所述持續整合伺服器選取的函式識別以及獲取的運行時間可以如下：

```

android.os.Handler.dispatchMessage (2061 ms);
android.os.Handler.handleCallback (2061 ms);
com.alipay.mobile.nebulacore.bridge.H5BridgeImpl$2.run (2061 ms);
com.alipay.mobile.nebulacore.bridge.H5BridgeImpl.access$100 (2061 ms);
com.alipay.mobile.nebulacore.bridge.H5BridgeImpl.executeWeb (2061 ms);
com.alibaba.fastjson.JSON.toJSONString (783 ms);
com.alibaba.fastjson.serializer.JSONSerializer.write (783 ms);
com.alibaba.fastjson.serializer.MapSerializer.write (783 ms);
com.alibaba.fastjson.serializer.MapSerializer.write (783 ms);
com.alibaba.fastjson.serializer.ListSerializer.write (783 ms);

```

com.alibaba.fastjson.serializer.ListSerializer.write (773 ms);
 com.alibaba.fastjson.serializer.MapSerializer.write (252 ms);
 android.os.Handler.dispatchMessage (2061 ms);
 android.os.Handler.handleCallback (2061 ms);
 com.alipay.mobile.nebulacore.bridge.H5BridgeImpl\$2.run (2061 ms);
 com.alipay.mobile.nebulacore.bridge.H5BridgeImpl.access\$100 (2061 ms);
 com.alipay.mobile.nebulacore.bridge.H5BridgeImpl.executeWeb (2061 ms);
 com.alibaba.fastjson.JSON.toJSONString (783 ms);
 com.alibaba.fastjson.serializer.JSONSerializer.write (783 ms);
 com.alibaba.fastjson.serializer.MapSerializer.write (783 ms);
 com.alibaba.fastjson.serializer.MapSerializer.write (783 ms);
 com.alibaba.fastjson.serializer.ListSerializer.write (783 ms);
 com.alibaba.fastjson.serializer.ListSerializer.write (773 ms);
 com.alibaba.fastjson.serializer.MapSerializer.write (124 ms)。

其中，函式識別 android.os.Handler.dispatchMessage、
 android.os.Handler.handleCallback、
 com.alipay.mobile.nebulacore.bridge.H5BridgeImpl\$2.run、
 com.alipay.mobile.nebulacore.bridge.H5BridgeImpl.access\$100、
 com.alipay.mobile.nebulacore.bridge.H5BridgeImpl.executeWeb、
 com.alibaba.fastjson.JSON.toJSONString、
 com.alibaba.fastjson.serializer.JSONSerializer.write (783 ms)、
 com.alibaba.fastjson.serializer.MapSerializer.write、
 com.alibaba.fastjson.serializer.MapSerializer.write、
 com.alibaba.fastjson.serializer.ListSerializer.write、
 com.alibaba.fastjson.serializer.ListSerializer.write、以及
 com.alibaba.fastjson.serializer.MapSerializer.write能夠形成一個子函式
 呼叫鏈。函式識別 android.os.Handler.dispatchMessage、

android.os.Handler.handleCallback、
com.alipay.mobile.nebulacore.bridge.H5BridgeImpl\$2.run、
com.alipay.mobile.nebulacore.bridge.H5BridgeImpl.access\$100、
com.alipay.mobile.nebulacore.bridge.H5BridgeImpl.executeWeb、
com.alibaba.fastjson.JSON.toJSONString、
com.alibaba.fastjson.serializer.JSONSerializer.write、
com.alibaba.fastjson.serializer.MapSerializer.write、
com.alibaba.fastjson.serializer.MapSerializer.write、
com.alibaba.fastjson.serializer.ListSerializer.write、
com.alibaba.fastjson.serializer.ListSerializer.write、以及
com.alibaba.fastjson.serializer.MapSerializer.write能夠形成一個子函式
呼叫鏈。

以下介紹本實施例中函式選取方法的一個具有應用場景。值得注意的是，本場景示例僅是為了更好地說明本實施例，並不構成對本實施例的不當限定。

在本場景示例中，在第一時刻，版本伺服器可以向持續整合伺服器發送應用程式第一版本的原始碼。所述持續整合伺服器可以接收所述應用程式第一版本的原始碼，所述應用程式第一版本的原始碼中可以包括除錯碼；可以編譯所述應用程式第一版本的原始碼，得到所述應用程式的第一版本；可以運行所述應用程式的第一版本，得到所述應用程式的第一版本在啟動階段的運行狀態資訊；可以分析啟動相關執行緒在所述運行狀態資訊對應的函式呼叫資訊，得到至少一個函式呼叫鏈；可以從所述至少一個函式

呼叫鏈中選取對應的運行時間大於或等於預設時間閾值的函式識別；可以將選取的函式識別識別的函式作為耗時函式。

在本場景示例中，在後續的第二時刻，版本伺服器可以向持續整合伺服器發送應用程式第二版本的原始碼。所述持續整合伺服器可以接收所述應用程式第二版本的原始碼，所述應用程式第二版本的原始碼中可以包括除錯碼；可以編譯所述應用程式第二版本的原始碼，得到所述應用程式的第二版本；可以運行所述應用程式的第二版本，得到所述應用程式的第二版本在啟動階段的運行狀態資訊；可以分析啟動相關執行緒在所述運行狀態資訊對應的函式呼叫資訊，得到至少一個函式呼叫鏈；可以從所述至少一個函式呼叫鏈中選取對應的運行時間大於或等於預設時間閾值的函式識別；可以將選取的函式識別識別的函式作為耗時函式。

在本場景示例中，所述持續整合伺服器可以將所述應用程式第一版本中的耗時函式與所述應用程式第二版本中的耗時函式進行比對；可以輸出所述應用程式第二版本中新增加的耗時函式，或者，可以輸出在所述應用程式第二版本中運行時間變長的耗時函式。這樣開發人員能夠發現所述應用程式第二版本原始碼中的缺陷，便於對所述應用程式第二版本的原始碼進行優化。

在本實施例中，所述持續整合伺服器可以運行原始碼中包括除錯碼的應用程式，得到所述應用程式的運行狀態

資訊；可以分析預設執行緒識別對應的函式呼叫資訊，得到所述預設執行緒識別對應的至少一個函式呼叫鏈；可以從所述至少一個函式呼叫鏈中選取對應的運行時間大於或等於預設時間閾值的函式識別。在本實施例中，所述持續整合伺服器可以分析應用程式中預設執行緒識別對應的函式呼叫資訊，得到至少一個函式呼叫鏈；可以基於預設時間閾值從所述至少一個函式呼叫鏈中過濾出耗時函式。這樣，所述持續整合伺服器可以提高耗時函式識別的效率和準確性。

請參閱圖 3。本說明書實施例還提供一種伺服器。所述伺服器可以包括以下單元。

運行單元 20，用於運行原始碼中包括除錯碼的應用程式，得到所述應用程式的運行狀態資訊；所述運行狀態資訊包括執行緒識別對應的函式呼叫資訊；所述函式呼叫資訊包括函式識別以及函式識別對應的時間戳；

分析單元 22，用於分析預設執行緒識別對應的函式呼叫資訊，得到所述預設執行緒識別對應的至少一個函式呼叫鏈；每個所述函式呼叫鏈包括至少一個函式識別；

選取單元 24，用於從所述至少一個函式呼叫鏈中選取對應的運行時間大於或等於預設時間閾值的函式識別；函式識別對應的運行時間基於該函式識別對應的時間戳得到。

請參閱圖 4。本說明書實施例還提供另一種伺服器。所述伺服器可以包括記憶體和處理器。

在本實施例中，所述記憶體包括但不限於動態隨機存取記憶體（Dynamic Random Access Memory，DRAM）和靜態隨機存取記憶體（Static Random Access Memory，SRAM）等。所述記憶體可以用於儲存電腦指令。

在本實施例中，所述處理器可以按任何適當的方式實現。例如，所述處理器可以採取例如微處理器或處理器以及儲存可由該（微）處理器執行的電腦可讀程式碼（例如軟體或韌體）的電腦可讀媒體、邏輯閘、開關、專用積體電路（Application Specific Integrated Circuit，ASIC）、可程式化邏輯控制器和嵌入微控制器的形式等等。所述處理器可以用於執行所述電腦指令實現以下步驟：運行原始碼中包括除錯碼的應用程式，得到所述應用程式的運行狀態資訊；所述運行狀態資訊包括執行緒識別對應的函式呼叫資訊；所述函式呼叫資訊包括函式識別以及函式識別對應的時間戳；分析預設執行緒識別對應的函式呼叫資訊，得到所述預設執行緒識別對應的至少一個函式呼叫鏈；每個所述函式呼叫鏈包括至少一個函式識別；從所述至少一個函式呼叫鏈中選取對應的運行時間大於或等於預設時間閾值的函式識別；函式識別對應的運行時間基於該函式識別對應的時間戳得到。

需要說明的是，本申請說明書中各個實施例均採用遞進的方式描述，各個實施例之間相同相似的部分互相參見即可，每個實施例重點說明的都是與其他實施例的不同之處。尤其，對於伺服器實施例而言，由於其基本相似於函

式選取方法實施例，所以描述的比較簡單，相關之處參見函式選取方法實施例的部分說明即可。

另外，本領域技術人員應當能夠理解的是，所屬領域技術人員在閱讀完本申請說明書之後，可以無需創造性勞動想到本申請檔案中列舉的部分或全部實施方式之間可以組合，這些組合也在本申請公開和保護的範圍內。

在20世紀90年代，對於一個技術的改進可以很明顯地區分是硬體上的改進（例如，對二極體、電晶體、開關等電路結構的改進）還是軟體上的改進（對於方法流程的改進）。然而，隨著技術的發展，當今的很多方法流程的改進已經可以視為硬體電路結構的直接改進。設計人員幾乎都透過將改進的方法流程程式化到硬體電路中來得到相應的硬體電路結構。因此，不能說一個方法流程的改進就不能用硬體實體模組來實現。例如，可程式化邏輯裝置（Programmable Logic Device, PLD）（例如現場可程式化閘陣列（Field Programmable Gate Array, FPGA））就是這樣一種積體電路，其邏輯功能由用戶對裝置程式化來確定。由設計人員自行程式化來把一個數位系統“整合”在一片PLD上，而不需要請晶片製造廠商來設計和製作專用的積體電路晶片。而且，如今，取代手工地製作積體電路晶片，這種程式化也多半改用“邏輯編譯器（logic compiler）”軟體來實現，它與程式開發撰寫時所用的軟體編譯器相類似，而要編譯之前的原始碼也得用特定的程式化語言來撰寫，此稱之為硬體描述語言（Hardware

Description Language, HDL), 而 HDL 也並非僅有一種, 而是有許多種, 如 ABEL (Advanced Boolean Expression Language)、AHDL (Altera Hardware Description Language)、Confluence、CUPL (Cornell University Programming Language)、HDCal、JHDL (Java Hardware Description Language)、Lava、Lola、MyHDL、PALASM、RHDL (Ruby Hardware Description Language) 等, 目前最普遍使用的是 VHDL (Very-High-Speed Integrated Circuit Hardware Description Language) 與 Verilog2。本領域技術人員也應該清楚, 只需要將方法流程用上述幾種硬體描述語言稍作邏輯程式化並程式化到積體電路中, 就可以很容易得到實現該邏輯方法流程的硬體電路。

上述實施例闡明的系統、裝置、模組或單元, 具體可以由電腦晶片或實體實現, 或者由具有某種功能的產品來實現。

上述實施例闡明的系統、裝置、模組或單元, 具體可以由電腦晶片或實體實現, 或者由具有某種功能的產品來實現。一種典型的實現設備為電腦。具體的, 電腦例如可以為個人電腦、膝上型電腦、蜂巢式電話、相機電話、智慧型電話、個人數位助理、媒體播放器、導航設備、電子郵件設備、遊戲主機、平板電腦、可穿戴設備或者這些設備中的任何設備的組合。

透過以上的實施方式的描述可知, 本領域的技術人員

可以清楚地瞭解到本說明書可借助軟體加必需的通用硬體平台的方式來實現。基於這樣的理解，本說明書的技術方案本質上或者說對現有技術做出貢獻的部分可以以軟體產品的形式體現出來，該電腦軟體產品可以儲存在儲存媒體中，如ROM/RAM、磁碟、光碟等，包括若干指令用以使得一台電腦設備（可以是個人電腦，伺服器，或者網路設備等）執行本說明書各個實施例或者實施例的某些部分所述的方法。

本說明書可用於眾多通用或專用的電腦系統環境或配置中。例如：個人電腦、伺服器電腦、手持設備或可攜式設備、平板型設備、多處理器系統、基於微處理器的系統、機上盒、可程式化的消費電子設備、網路PC、小型電腦、大型電腦、包括以上任何系統或設備的分散式計算環境等等。

本說明書可以在由電腦執行的電腦可執行指令的一般上下文中描述，例如程式模組。一般地，程式模組包括執行特定任務或實現特定抽象資料類型的常式、程式、物件、組件、資料結構等等。也可以在分散式計算環境中實踐本說明書，在這些分散式計算環境中，由透過通信網路而被連接的遠端處理設備來執行任務。在分散式計算環境中，程式模組可以位於包括儲存設備在內的本地和遠端電腦儲存媒體中。

雖然透過實施例描繪了本說明書，本領域普通技術人員知道，本說明書有許多變形和變化而不脫離本說明書的

精神，希望所附的申請專利範圍包括這些變形和變化而不脫離本說明書的精神。

【符號說明】

S10：步驟

S12：步驟

S14：步驟

20：運行單元

22：分析單元

24：選取單元



I684916

公告本

【發明摘要】

【中文發明名稱】

函式選取方法和伺服器

【中文】

本說明書實施例提供一種函式選取方法和伺服器。所述方法包括：運行原始碼中包括除錯碼的應用程式，得到所述應用程式的運行狀態資訊；所述運行狀態資訊包括執行緒識別對應的函式呼叫資訊；所述函式呼叫資訊包括函式識別以及函式識別對應的時間戳；分析預設執行緒識別對應的函式呼叫資訊，得到所述預設執行緒識別對應的至少一個函式呼叫鏈；每個所述函式呼叫鏈包括至少一個函式識別；從所述至少一個函式呼叫鏈中選取對應的運行時間大於或等於預設時間閾值的函式識別；函式識別對應的運行時間基於該函式識別對應的時間戳得到。

【指定代表圖】第(2)圖。

【代表圖之符號簡單說明】無

【特徵化學式】無

【發明申請專利範圍】

【第1項】

一種函式選取方法，包括：

運行原始碼中包括除錯碼的應用程式，得到所述應用程式的運行狀態資訊；所述運行狀態資訊包括執行緒識別對應的函式呼叫資訊；所述函式呼叫資訊包括函式識別以及函式識別對應的時間戳；

分析預設執行緒識別對應的函式呼叫資訊，得到所述預設執行緒識別對應的至少一個函式呼叫鏈；每個所述函式呼叫鏈包括至少一個函式識別；以及

從所述至少一個函式呼叫鏈中選取對應的運行時間大於或等於預設時間閾值的函式識別；函式識別對應的運行時間基於該函式識別對應的時間戳得到。

【第2項】

如請求項1所述的方法，函式識別對應的時間戳包括進入時間戳和退出時間戳。

【第3項】

如請求項2所述的方法，函式識別對應的運行時間採用如下方式計算得到：

以函式識別對應的退出時間戳表示的時刻為退出時刻，以該函式識別對應的進入時間戳表示的時刻為進入時刻，計算所述退出時刻和所述進入時刻之間的差值，作為該函式識別對應的運行時間。

【第4項】

如請求項1所述的方法，所述運行狀態資訊包括所述應用程式在預設運行階段的運行狀態資訊；所述預設運行階段包括啟動階段。

【第5項】

如請求項1所述的方法，所述運行原始碼中包括除錯碼的應用程式，得到所述應用程式的運行狀態資訊，包括：

接收版本伺服器發來的應用程式的原始碼；所述原始碼中包括除錯碼；

基於所述原始碼，產生所述應用程式；以及

運行所述應用程式，得到所述應用程式的運行狀態資訊。

【第6項】

如請求項1所述的方法，所述分析預設執行緒識別對應的函式呼叫資訊，得到所述預設執行緒識別對應的至少一個函式呼叫鏈，包括：

獲取預設執行緒識別在所述運行狀態資訊中對應的函式呼叫資訊；以及

以獲取的函式呼叫資訊為目標函式呼叫資訊，分析所述目標函式呼叫資訊中各個函式識別識別的函式間的呼叫關係，得到所述預設執行緒識別對應的至少一個函式呼叫鏈。

【第7項】

如請求項1所述的方法，所述從所述至少一個函式呼

叫鏈中選取對應的運行時間大於或等於預設時間閾值的函式識別，包括：

針對所述至少一個函式呼叫鏈中的每個函式呼叫鏈，在該函式呼叫鏈包括對應的運行時間大於或等於預設時間閾值的函式識別時，從該函式呼叫鏈中選取對應的運行時間大於或等於預設時間閾值的函式識別；選取的函式識別能夠形成該函式呼叫鏈的一個子函式呼叫鏈。

【第8項】

如請求項1所述的方法，所述方法還包括：

獲取選取的函式識別所對應的運行時間。

【第9項】

一種伺服器，包括：

運行單元，用於運行原始碼中包括除錯碼的應用程式，得到所述應用程式的運行狀態資訊；所述運行狀態資訊包括執行緒識別對應的函式呼叫資訊；所述函式呼叫資訊包括函式識別以及函式識別對應的時間戳；

分析單元，用於分析預設執行緒識別對應的函式呼叫資訊，得到所述預設執行緒識別對應的至少一個函式呼叫鏈；每個所述函式呼叫鏈包括至少一個函式識別；以及

選取單元，用於從所述至少一個函式呼叫鏈中選取對應的運行時間大於或等於預設時間閾值的函式識別；函式識別對應的運行時間基於該函式識別對應的時間戳得到。

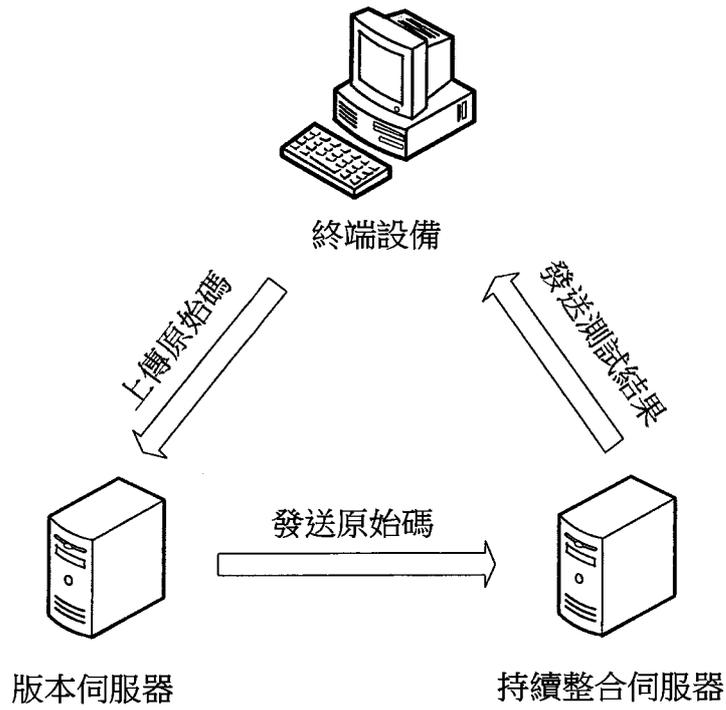
【第10項】

一種伺服器，包括：

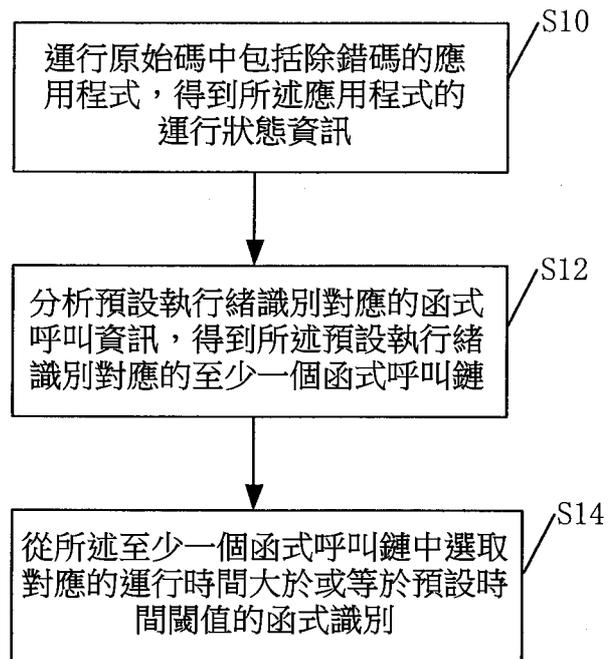
記憶體，用於儲存電腦指令；以及

處理器，用於執行所述電腦指令實現以下步驟：運行原始碼中包括除錯碼的應用程式，得到所述應用程式的運行狀態資訊；所述運行狀態資訊包括執行緒識別對應的函式呼叫資訊；所述函式呼叫資訊包括函式識別以及函式識別對應的時間戳；分析預設執行緒識別對應的函式呼叫資訊，得到所述預設執行緒識別對應的至少一個函式呼叫鏈；每個所述函式呼叫鏈包括至少一個函式識別；從所述至少一個函式呼叫鏈中選取對應的運行時間大於或等於預設時間閾值的函式識別；函式識別對應的運行時間基於該函式識別對應的時間戳得到。

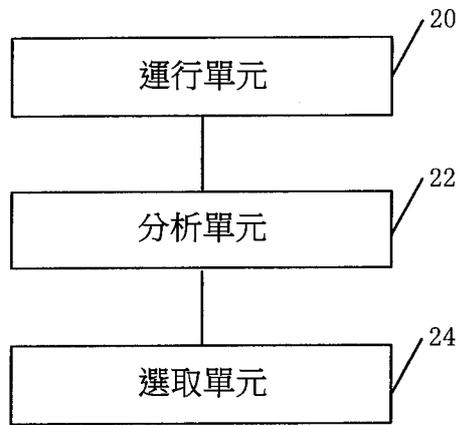
【發明圖式】



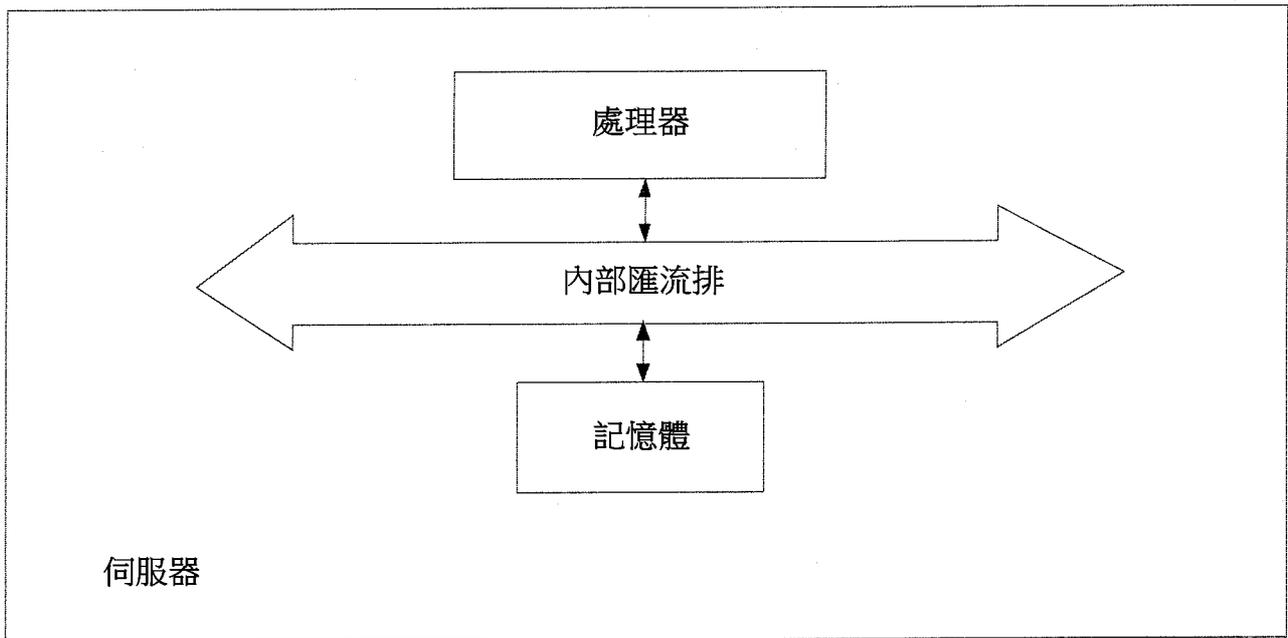
【圖 1】



【圖 2】



【圖 3】



【圖 4】